SQL SELECT

SQL DISTINCT

SQL WHERE

SQL AND OR

SQL IN

SQL BETWEEN

SQL 萬用字元

SQL LIKE

SQL ORDER BY

SQL 函數

SQL 平均值

SQL COUNT

SQL 最大值

SQL 最小值

SQL 總合

SQL GROUP BY

SQL HAVING

SQL 別名

SQL AS

SQL 表格連接

SQL 外部連接

SQL CONCATENATE

SQL SUBSTRING

SQL TRIM

SQL 長度

SQL REPLACE

SQL CREATE TABLE

SQL Constraint

SQL 主鍵

SQL 外來鍵

SQL CREATE VIEW

SQL CREATE INDEX

SQL ALTER TABLE

SQL DROP TABLE

SQL TRUNCATE TABLE

SQL INSERT INTO

SQL UPDATE

SQL DELETE FROM

https://i.imgur.com/U92Q9ms.gif

```
C:\Users\品瑜\Desktop\fibo_iterative.exe
0,1,1,2,3,5,8,13,21,34,55,89,144,233,
--------------------------------
Process exited after 0.0854 seconds with return value 0
請按任意鍵繼續 . . .
```

```
#include<stdio.h>

int Fibo(int n)
{
    if(n==0)
    {
```

```
        return 0;
    }
    else if(n==1)
    {
        return 1;
    }
    else
    {
        int a = 0;
        int b = 1;
        int c;
        for(int i=1;i<n;i++)
        {
            c = a + b;
            a = b;
            b = c;
        }
        return c;
    }
}
int main()
{
    for(int i=0;i<=13;i++)
    {
        printf("%d,",Fibo(i));
    }
    return 0;
}
```

Time Complexity : O(n)

```
C:\Users\品瑜\Desktop\Fibo_recursive.exe
0,1,1,2,3,5,8,13,21,34,55,89,144,233,
--------------------------------
Process exited after 0.03367 seconds with return value 0
請按任意鍵繼續 . . .
```

```
#include<stdio.h>
```

```
int Fibo(int n)
{
    if(n==0)
    {
        return 0;
    }
    else if(n==1)
    {
        return 1;
    }
    else
    {
        return Fibo(n-1)+Fibo(n-2);
    }
}


int main()
{
    for(int i=0;i<=13;i++)
    {
        printf("%d,",Fibo(i));
    }
    return 0;
}
```

Time Complexity : O(2^n)

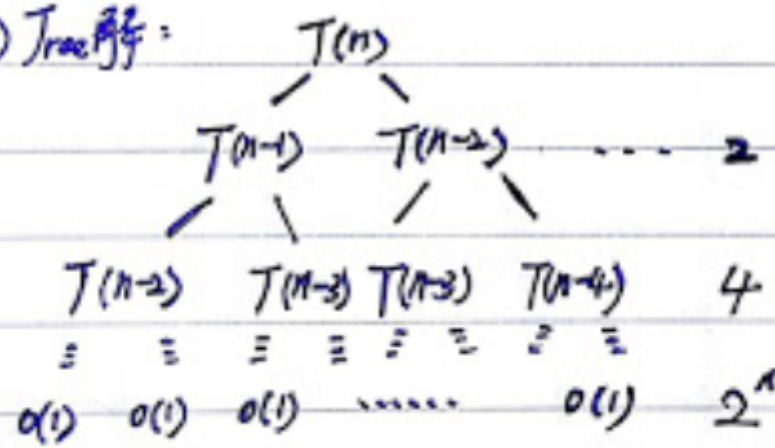
(1) $T(0) = O(1)$, $T(1) = O(1)$

$$T(n) = T(n-1) + T(n-2) + O(1)$$

$$= O(2^{n-1}) + O(2^{n-2}) + O(1) = O(2^n) \text{ ... exp. time}$$

(2) 另解：$F_n = \frac{1}{\sqrt{5}} \cdot \left[\frac{1}{2}(1+\sqrt{5})\right]^n = \Theta(1.6^n)$ ... exp. time

(3) Tree解：



$$\therefore T(n) = 2 + 4 + \cdots + 2^n = O(2^n) \text{ ... exp. time}$$

C:\Users\品瑜\Desktop\Fibo_DP.exe

```
0,1,1,2,3,5,8,13,21,34,55,89,144,233,
--------------------------------
Process exited after 0.03586 seconds with return value 0
請按任意鍵繼續 . . .
```

```
#include<stdio.h>





int Fibo(int n)
{
   int F[n+1];
   F[0] = 0;
   F[1] = 1;
   if(n<2)
   {
        printf("0,1");
   }
   else
   {
        for(int i=2;i<=n;i++)
        {
          F[i] = F[i-1] + F[i-2];
     }
   }
   for(int k=0;k<=n;k++)
   {

     printf("%d,",F[k]);
   }
}


int main()
```

```
{
    Fibo(13);
    return 0;
}
```

Time Complexity : O(n)

```
#include <stdio.h>
int count = 0;
void hanoi(int n, char A, char B, char C)
{
    if (n == 1)
    {
        printf("%d: 將第 %d 個圓盤由 %c 移到 %c\n", count++, n, A, C);
    }
    else
    {
        hanoi(n - 1, A, C, B);
        printf("%d: 將第 %d 個圓盤由 %c 移到 %c\n", count++, n, A, C);
        hanoi(n - 1, B, A, C);
    }
}
int main()
{
    int n;
    printf("請輸入河內塔的高度：");
    scanf("%d", &n);
    hanoi(n, 'A', 'B', 'C');
    printf("移動 %d 層河內塔共需移動 %d 次\n", n, count);
    return 0;
}
```

T(n) = T(n-1) + T(1) + T(n-1)，且 T(1) = 1 ；則 T(n) = 2*T(n-1) + T(1) ，解出 T(n) 為 (2^n) -1。

排列

#include<stdio.h>

void swap(char* pch,char *pBegin)

{

char temp = *pch;

*pch = *pBegin;

*pBegin = temp;

```
}
void Perm(char list[], int i, int n){ //list[i] ~ list[n] permutation
  int j;
  if(i == n){
    for (j = 0; j <= n; j++){
      printf("%c ", list[j]);
    }
    printf("\n");
  }
  else{ //i < n
      for(j = i; j <= n; j++){
        swap(&list[i], &list[j]); //list[j] as head
        Perm(list, i+1, n); //list[i+1] ~ list[n] permutation
        swap(&list[i], &list[j]); //return to the original list
      }
  }
}


int main()
{
char list[] = "abcd";
Perm(list,0,3);
return 0;
}
```